5-29-2003

Discovery, Visualization and Analysis of Gene Regulatory Sequence Elements in Genomes

Daniel F. Simola
Dartmouth College

Daniel F. Simola
Dartmouth College
Computer Science Technical Report TR2003-456
Computer Science Honors Senior Thesis:
Discovery, Visualization and Analysis of Gene Regulatory Sequence Elements in Genomes
5/29/03

Faculty Committee:
Bob Gross, Scot Drysdale, Jay Aslam (advisor)

Table of Contents

List of Figures

List of Tables

Abstract

The advent of rapid DNA sequencing has produced an explosion in the amount of available sequence information, permitting us to ask many new questions about DNA. There is a pressing need to design algorithms that can provide answers to questions related to the control of gene expression, and thus to the structure, function, and behavior of organisms. Such algorithms must filter through massive amounts of informational noise to identify meaningful conserved regulatory DNA sequence elements.

We are approaching these questions with the notion that visualization is a key to exploring data relationships. Understanding the exact nature of these relationships can be very difficult by simply interpreting raw data. The ability to look at data in a graphical form allows us to apply our innate capacity to think visually to discern the subtle relationships that might not be recognizable otherwise.

This thesis provides computational tools to visually identify and analyze candidate motifs in the DNA of a species. This includes a parsing utility to store genomic data and an application to search for and visually identify motifs. Using these tools, novel and previously compiled gene sets were identified using the genome of the plant species *Arabidopsis thaliana*.

I – Introduction

In this thesis I wish to provide a suite of tools for biologists to identify, visualize, and analyze significant DNA sequence elements in a genome. This thesis is one of bioinformatic relevance; it is written for biologists and computer scientists alike, as it involves information processing, biological modeling, algorithm design, database manipulation, and an overall understanding of biological and computational systems. This research has focused on the ability to ask questions about the relevance of DNA sequence elements in gene regulation.

Regulation of gene expression may occur at several stages before a final protein is produced. We, however, are focusing on one specific type of regulation, the promoter-based regulation of gene transcription[1]. The two regions most likely to affect the level of transcription are the promoter and the enhancer. The enhancer region is broadly defined as a sequence of DNA of varying distance from a promoter, which can function on either strand of DNA to affect a promoter's activity[2]. In reality enhancer sequences thousands of bases upstream (or even downstream) of the start site for a gene can affect transcription. Thus an enhancer region is very ambiguous and difficult to identify. A region more conducive to bioinformatics investigation is the upstream promoter region, which usually encompasses a few hundred bases upstream of the beginning of a gene. We have chosen to look at sequences extending 1,500 bases upstream. This range is most likely to contain conserved DNA elements that affect gene regulation, since the key place for regulation is the promoter itself – the start site of transcription and location for the formation of the promoter complex of proteins, known as the basal transcription

apparatus. Although there exist regulatory elements outside of this 1,500 base region, they represent a small minority of regulatory elements. Our analysis will not identify these regions.

A DNA sequence element is sometimes called a motif, many of which are highly conserved. By definition the nucleotides comprising a genome are not randomly ordered. Presumably then, important regulatory motifs exist due to the forces of natural selection, which have favored the conservation of certain sequences and positions and not influenced those of others. Thus natural selection creates order from randomness. Knowing this, one expects to find certain motifs (or subsets thereof) more frequently and in specific locations than others in a genome. By the same token, there are also many important motifs that occur very rarely. Natural selection can favor either, but finding a motif that differs greatly from the mean does not imply importance alone. For example, there are numerous repeated sequences that provide little specific information content, but have existed and multiplied via idiosyncrasies in DNA replication and crossing over. Such sequences are found in so many copies that they cannot provide meaningful functionality. Rare motifs can also exist simply because they do not affect the genome and so are not selected out of it. Thus to identify a significant motif, one must consider both its frequency and location. Let us then define significance as the quality of a motif possessing certain unique, observable characteristics. These characteristics may include its position, sequence, or orientation, for example. That is, we consider a motif significant when it is relevant to the promoter-based regulation of gene transcription.

In considering motifs of different lengths, many copies of shorter motifs are more likely to occur in a genome just through chance, and thus analysis of data based on these motifs provides more noise to filter. In other words, since the information content of, say a 4-mer, is relatively low, you would need to see many occurrences of a given motif to consider it significant (e.g. if a motif occurs 500 times, but every time it is found exactly 126 bp from the transcription start, that would be significant). As motif length increases, the chance of finding a particular motif randomly decreases, and thus the possibility of identifying a unique sequence increases. However this does not necessarily imply that significance increases with motif length.

Biology provides even more constraints. Finding multiple occurrences of a motif, while statistically satisfying, does not imply biological relevance. It is not necessarily meaningful to find a common motif distributed randomly (or uniformly) throughout a genome. Given our knowledge of genetics, it would be much more pleasing if this motif occurred often in the same relative location, such as always in the promoter region. A motif is biologically significant when it has both a defined sequence and position (in one or more genes); that is, significance derives from evolutionary conservation. Throughout this paper then, the term significant refers specifically to motifs that have non-random distribution and defined locations. These motifs may not be biologically significant (i.e. they might not be involved in transcriptional regulation), but are nevertheless good candidates of true biological importance.

For example, there is a motif called a ‘TATA-box’ found in the promoter region of most genes in every eukaryotic organism. It serves as a crucial positioning component of the promoter[3]. The conserved sequence of the TATA-box is actually eight bases long, and is usually located about 25 bases upstream of the transcription start point[4]. Location is what gives the TATA-box its importance. If this sequence were located anywhere else

in the promoter, transcription would not occur. To further support the notion that significant motifs bear high information content, single base mutations in the TATA-box serve as strong down mutations. Thus a mutation in the TATA-box alters the startpoint for transcription initiation, changing the function of the motif, as encoded by its base sequence.

Such discoveries result from the interest there is in investigating the existence of conserved motifs, and this same interest urges for the investigation of other promoter-based motifs. One certainty is that there is no shortage of sequence data to examine. Several dozen genomes have been sequenced completely, and eventually databases will contain as many genomes as scientists care to sequence. The key question, then, is how to identify a significant sequence. To answer this question, a suite of tools has been developed. This suite includes a relational database, containing annotated sequence data for *Arabidopsis thaliana*, and will soon include four other species; Genome Parser, a general parsing tool to extract and format relevant information from GenBank formatted sequence files to import into the database; VxInsight, a commercial application which provides visualization tools for analyzing similarities between genes; and Motif Browser, an application which identifies and graphically displays motifs within a set of genes. This thesis discusses each tool, however I developed Genome Parser and Motif Browser, while facilitating the ability of VxInsight to connect with our database.

The Bob Gross Laboratory (BGLab) is approaching this research with the notion that visualization is a key to exploring data relationships. Understanding the exact nature of these relationships can be very difficult by simply interpreting raw data. The ability to look at data in a graphical form allows us to apply our innate capacity to think visually to discern the subtle relationships that might not be recognizable otherwise. To this end, the database and parser provide the means to retrieve and store data; VxInsight and Motif Browser provide visual representations of the data. The goal is to use this suite to identify significant genomic sequences.

In review, this thesis aims to provide tools to visually identify and analyze candidate motifs in the DNA of a species. Using these tools, novel and previously compiled gene sets were studied in order to determine any biological function. Results of this research could provide insights into answering fundamental questions about gene expression, and could have wide-ranging effects from enhancing our concept of genetics to learning how to treat human disorders.

Background

The biologist's grail is to fully reverse engineer a species' genome, to understand how an organism develops and maintains itself, and to be able to manipulate, and hopefully one day, improve upon this great foundation. It is no surprise then, that geneticists, molecular biologists, and biochemists, along with those in other disciplines of biology, have traditionally focused their attention on the building blocks of life, namely nucleic acids and amino acids. These molecules combine to form more complex forms, such as deoxyribonucleic acid (DNA), ribonucleic acid (RNA), and polypeptides (protein). The central dogma of biology states that genes are perpetuated through nucleic acids, and function is achieved through gene expression in the form of proteins. Transcription and translation are responsible for converting between forms[5]. Since this

dogma's inception, the quest at least implicitly began to map an organism's genome, the set of all genes of an organism. Even James Watson, discoverer of the helical structure of DNA, realized this goal; he was the Human Genome Project's main advocate and first director[6].

The quest does not end with sequenced genomes, however. Afterward there is the task of compiling transcriptomes (set of all RNA transcripts) and proteomes (set of all proteins), not to mention the work involved in deciphering and understanding this vast sea of data. Recent advances in technology have provided the means to rapidly sequence genetic information. Thus, in the past decade there has been a directed effort to completely sequence the genomes of certain organisms. The apex of genetic sequencing occurred in 2000, when the preliminary draft of the human genome was released.

But what is a genome really? Theoretically it is simply a large set of data, of information, in its most raw form. By definition information is inherently nonrandom. Random strings of DNA cannot explicitly contain meaning, as they are random. A genome can be considered as a long string of letters, of which there are four: adenine (A), thymine (T), cytosine (C), and guanine (G). It can be designed in an infinite number of ways. For any given sequence of n bases, there are theoretically 4^n possible genomes. In reality there are many fewer genomes than this, since any two species share some amount of sequence, but the notion is that the possible phase space for DNA is immense. Thus the vast diversity of life on Earth – for example, how many people in the world look identical?

Functionally a genome is a composition of chromosomes. Within a chromosome lie thousands of genes, each of which, when expressed, serves as a template for creating RNA and then protein (as mentioned above). A gene has an upstream region usually containing a promoter, which contains regulatory regions that enable the level of expression of the adjacent gene to be regulated; a terminal region, which specifies the end of the coding region; and introns and exons within the body of the gene. Introns are something like nonsense sequences within a coding sequence, while exons are the critical sequences, which compose the body of mature RNA. After transcription (the process of reading a gene and producing a strand of RNA), introns are spliced from a precursor mRNA in such as way as to produce a final mRNA. In many species, especially higher order organisms like humans, alternative splicing occurs in different tissues and is used to generate a diversity of gene products from a single region of DNA. Since only 3-5% of all genes are expressed in any cell at a given time, the regulators of expression directly determine cell behavior and guide such critical processes as development and immunity.

Many biologists share the task of examining DNA sequences to identify such genetic regulators. In order to examine these sequences, they must first be identified. That is the purpose of genome sequencing. The next step is the maintenance of and access to this information. The best storage medium today is the computer; the best way to store large amounts of information is in a database; and the best way of providing access to this information is over the Internet.

Biologists and computer scientists have quickly realized these facts, and a new interdisciplinary field called bioinformatics has emerged, combining aspects from genetics, mathematics, statistics, and computer science. Geneticists wish to solve the mysteries of life, while computer scientists provide the means to analyze the data necessary to approach that question, by developing databases and algorithms to facilitate

biological research. Indeed, bioinformatics provides an entirely new way of performing biological research. Not only can a scientist more quickly identify sequence information and filter out unwanted data or meaningless sequences, but he or she can also ask a completely different set of questions. Heavily computational processes such as creating phylogenetic trees, interspecies genome comparisons, and conserved motif searches are all very possible today and are in fact active areas of research. Software developers have provided crops of new tools to biologists, and biologists have provided new problems and puzzles to ponder. The fusion of computer science and biology has indeed blazed a new path for science.

II – The Database

Goals

The BGLab database was first launched in 2002 to help biologists discover and extract meaning from DNA sequence information. This database currently contains annotated sequence information from *Arabidopsis thaliana* to facilitate this research. Other genomes will be added in the future. Most genomes today are annotated in (what are finally becoming) standardized formats, such as GenBank, and freely available for download on the Internet. In addition to the raw sequence data, the annotations provide regional breakdowns of a genome, known as loci, genes, transcript and protein sequences and functional descriptions for them, among other things. At the level of the gene, annotations provide ranges for exons and introns. From these ranges, we can compute other significant ranges, including the up1500 region, the down500 region, and the untranslated regions (UTR). Of course, the information included in annotated sequence files is often hypothetical or experimental, and more often incomplete. This necessitates that the database be dynamic, and able to incur (relatively) frequent updating with fresh copies of genomic data.

In addition to raw sequence data, the database stores motifs of length 9 (9-mers) for every genome. Thus there is a 9-mer starting at essentially every base in a genome. There are 4^9 possible permutations of this motif (262,144). Using these 9-mers you can also examine motifs of shorter lengths. The length of nine was chosen primarily for one reason. Looking at any motif greater than 9 bases would not generate enough instances from a genome to provide us with interesting and meaningful statistics.

The lab's current efforts focus on the genome of *Arabidopsis thaliana*, a small flowering plant. *Arabidopsis* has 25,498 genes[7], and about 125 million nucleotides in its genome (see Table 1). Thus there are nearly 125 million 9-mers for *Arabidopsis*. Probabilistically, the longer the motif length is, the less likely it is to find a particular permutation in the set of motifs. Assuming a random distribution of motifs, the probability of finding a given 9-mer permutation is 3.815×10^{-6}. The expected number (mean) of occurrences of a given 9-mer in *Arabidopsis* is 477. The smallest motif we are using is the 4-mer. There are 256 possible 4-mer permutations, and the probability of finding a particular 4-mer is 0.0039. One would expect to find about 487,000 occurrences of a 4-mer in *Arabidopsis*. Hence the amount of expected noise resulting from calculations on this range of motif lengths varies greatly. If we looked at 10-mers, we

could only expect to find 119 instances of a motif in *Arabidopsis*. For this reason 9-mers were chosen as the maximum length motif. Of course larger genomes such as the human genome are long enough to generate meaningful 10-mer statistics. Perhaps 10-mers will be used in the future, but it is currently simpler and more prudent to standardize on one length for all genomes.

Database Structure

The BGLab database runs on Edison.Dartmouth.edu over port 2638. The database is currently running Sybase's Adaptive Server Anywhere software suite, version 7.0. The BGLab has focused on *Arabidopsis thaliana* as a prototype organism for implementing the database, but recently broadened its scope to incorporate four additional species. In the spirit of motif discovery and analysis, Jonathan Carlson '03 designed the schema of Figure 1, which was implemented in the Arabidopsis database launched in August 2002.

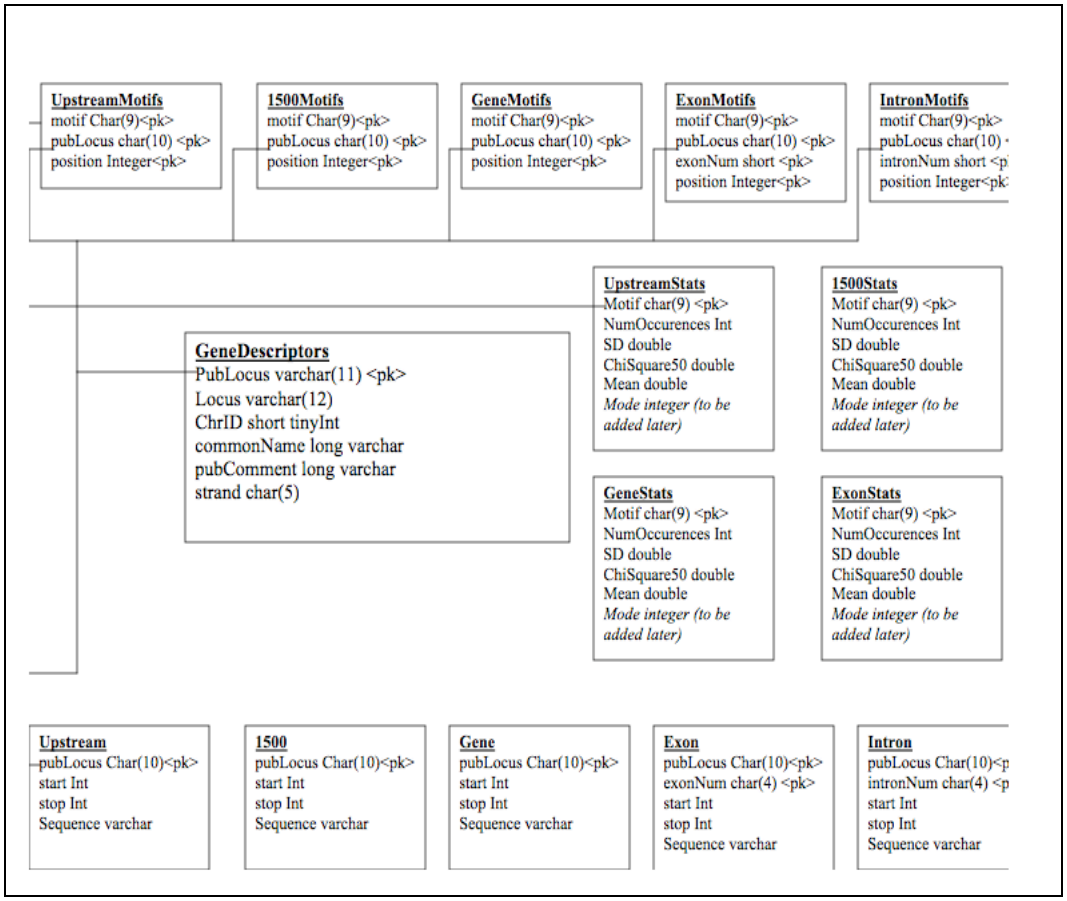

Figure 1: Arabidopsis Database Schema (courtesy of Jonathan Carlson)

This database stores all of the significant ranges of DNA for the *Arabidopsis thaliana* genome, as well as statistics computed on regions upstream of genes (both from the beginning of the first exon upstream 1,500 bases or and until the end of the last exon of the previous gene, whichever comes first, and from the beginning of the first exon

upstream 1500 bases, regardless), on genes, introns, and exons of genes. Statistics include number of occurrences, standard deviation, chi squared, and mean.

The Arabidopsis database has recently been revised both to improve upon its schema and to permit the addition of other genomes. In general, the schema now exhibits a more modular and scalable design. Most importantly the primary gene identifier is now an integer value, instead of the pub locus name (varchar). Some genomes are not as well organized as *Arabidopsis*, and it is not guaranteed that every gene name is unique. Integer keys are used to guarantee unique genes.

Two new statistics were added: KS D (the Kolmogorov-Smirnov statistic) and KS probability (the p-value associated with this statistic). The statistic itself is non-parametric and tests whether an unbinned distribution is from an expected distribution. That is, it tests whether a given distribution is significant. The p-value gives the probability that the distribution is significant[8]. Mean, mode, and number of occurrences of mode have also been added, along with other minor changes.

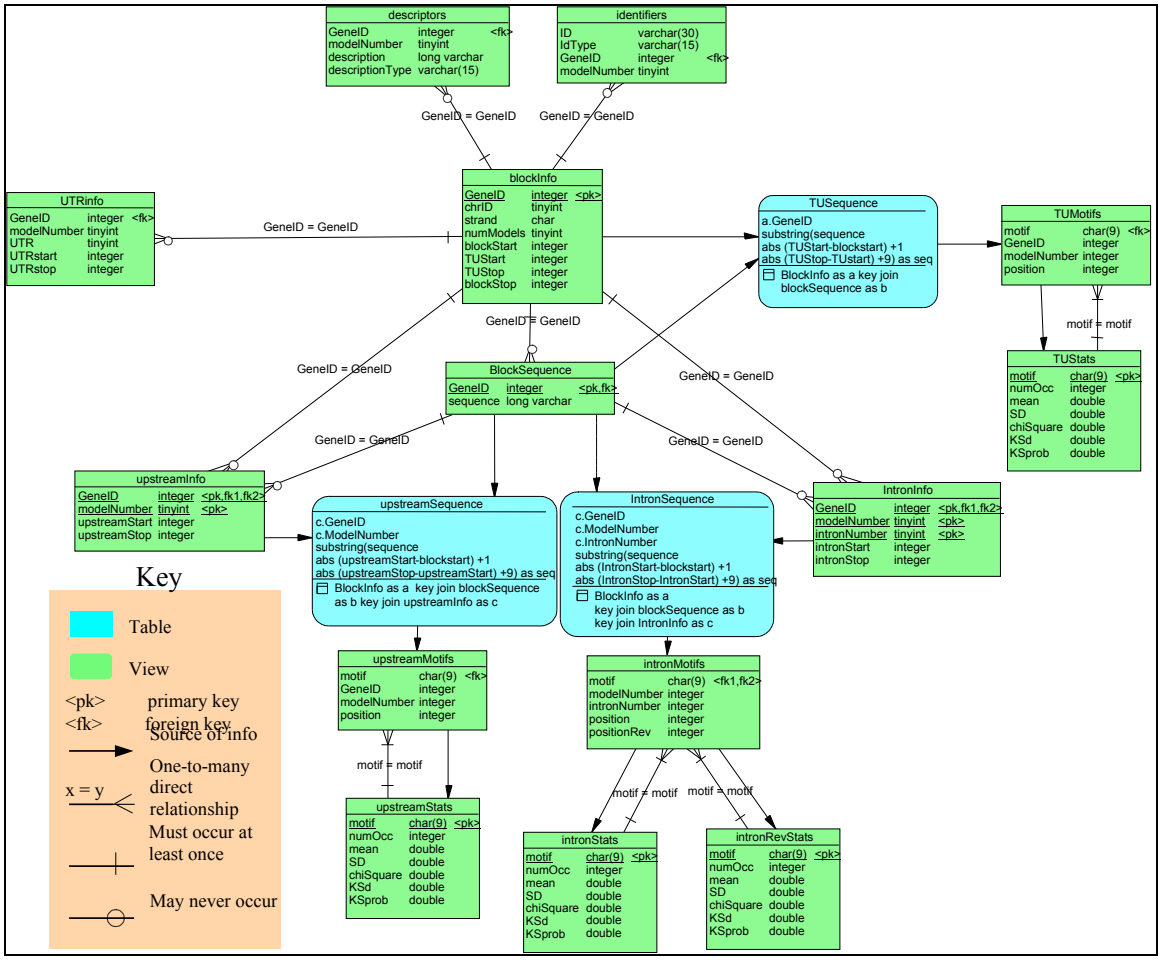

Figure 2: Genomes Database Schema (courtesy of Jonathan Carlson)

Model Organisms

The Genomes database will initially contain five species, each a model system in biology (excluding one): *Caenorhabditis elegans, Arabidopsis thaliana, Saccharomyces cerevisiae, Homo sapiens,* and *Drosophila melanogaster*. A model organism is one that can be easily observed and studied in a laboratory context[9]. By this definition the human species is not necessarily a model organism, however it is a model of higher order organisms due to its complexity and evolutionary status. There are usually only a handful of similar model organisms and are chosen because of some unique or otherwise valuable characteristic, or because they are related to another species. In other words, the choice is based on the characteristics of particular genetic sequences.

For example, *C. elegans* is a multicellular eukaryote used for studying growth and development. Scientists have been able to map the lineage of every cell from embryogenesis to mature adult. Geneticists favor *Drosophila* because of its "rapid life cycle, its readiness to breed, and the polytene chromosomes of the larva (which allow gene localization)"[10]. The *Arabidopsis* genome is the first completed plant sequence, and it provides a core set of plant-specific genes to study. *S. cerevisiae* is a model lower eukaryote (i.e. unicellular eukaryote). It has a high density of genes, meaning that the overall information content of the genome is high, and there is minimal randomness and repetition[11]. Of course, the human genome serves as the model for higher order multicellular eukaryotes. Table 1 below lists the number of genes and number of bases for each of the selected organisms above.

Organism	**Number of Bases (MB)**	**Number of Genes**
Arabidopsis thaliana	125	25,498
Caernorhabitis elegans	97	19,099
Drosophila melanogaster	165	13,700
Homo sapiens	3,300	~30,000
Saccharomyces cerevisiae	13.5	5,885+

Table 1: Genome Sizes and Gene Numbers[12]

As can be seen from Table 1, each organism varies with respect to gene content. For example *S. cerevisiae* contains roughly half the number of genes as *Drosophila*, but is contained in 13.5MB rather than 165MB. In addition to this content variation, each species occupies a unique position in the overall DNA phase space. That is, if you think of a genome as one long string of a few hundred mega bases, one genome can differ from another at any point along the length of the string. DNA sequences encode certain characteristics or functions for an organism. The theoretical set of all possible species characteristics is the DNA phase space. The differences in the genes of each organism place the organism in a different area of the phase space. For both of these reasons, these five genomes provide a great dataset for future research involving interspecies comparisons.

III – Genome Parser

Once the data from a genome is loaded into the Genomes database, it is possible to examine it. This is no small task, however. Initially the data had to be extracted from the appropriate source files and converted into a format to import into the database. Data that was not explicitly enclosed in the files had to be generated as well. We needed create a mechanism of converting GenBank formatted data for an organism into a format we could use to import directly into our database. The output of the parser is separated into several files, each of which corresponds to a table in the Genomes schema. I spent the whole of winter term, 2003 designing and implementing Genome Parser.

GenBank Formatted Files

It has taken some time, but the file formats used to store annotated genomic sequences are finally becoming standardized. This does not mean, however, that all sequence files downloadable today are of the same format. Discrepancies between files of the same format remain and will for some time. The primary formats for storing genomic data are FASTA, GenBank, and EMBL. Designers based the formats on the National Center for Biotechnology Information (NCBI) data model for sequence related information, released over a decade ago[13]. The goal of the NCBI was to create a model conducive to easy storage and retrieval of sequence information, as well as long-term stability. The ISO standard ASN.1 was used to implement this model, providing an easily accessible, cross-platform solution[14].

Both FASTA and GenBank formats store the sequence data, annotated with some descriptive information. Fasta and EMBL however do not usually contain much additional annotation, whereas GenBank offers a rich feature set. GenBank files contain regions called coding sequences (CDS), namely those that code for protein. The CDS annotation is known as a feature of the format. GenBank files usually offer several other features as well, further delimiting the CDS DNA entry. Primarily this includes Gene and mRNA features. Gene features are also commonly subdivided into exon features.

We chose to use GenBank formatted files as the source of data for the BGLab database. GenBank files are composed of data from the GenBank Genetic Sequence Data Bank, which shares its entries with the databases of the European Molecular Biology Laboratory (EMBL) and DNA Data Bank of Japan (DDBJ). Thus these three databases rely on a common format and source of data[15]. These files are freely available and downloadable from the NCBI website <http://www.ncbi.nlm.nih.gov/>. They provide a generally complete feature set for our needs.

A GenBank formatted file has the following overall structure. There is a header region, containing the Accession number, version, and GI number for a sequence. This sequence can either be a chromosome or a particular locus of a chromosome. The header is followed by the feature list. As mentioned above, the major features are Gene, mRNA, and CDS. Figure 3 contains a sample GenBank file. Appendix 1 part A contains more complete details regarding the GenBank format.

Genome Parser Requirements

```
sample genbank file                                                    Page 1 of 1
Printed: Friday, May 23, 2003 8:47:32 PM

LOCUS       NC_003070           30080809 bp    DNA     linear   PLN 10-JAN-2002
DEFINITION  Arabidopsis thaliana chromosome 1, complete sequence.
ACCESSION   NC_003070
VERSION     NC_003070.2  GI:18426880
KEYWORDS    HTG.
SOURCE      Arabidopsis thaliana
  ORGANISM  Arabidopsis thaliana
            Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
            Spermatophyta; Magnoliophyta; eudicotyledons; core eudicots;
            Rosidae; eurosids II; Brassicales; Brassicaceae; Arabidopsis.
REFERENCE   1  (bases 1 to 30080809)
  AUTHORS   Town,C.D., Haas,B.J., Wu,D., Maiti,R., Hannick,L.I., Chan,A.P.,
            Tallon,L.J., Rooney,T., Utterback,T.R., VanAken,S.E.,
            Feldblyum,T.V., White,O. and Fraser,C.M.
  TITLE     Arabidopsis thaliana chromosome 1 genomic sequence
  JOURNAL   Unpublished
REFERENCE   2  (bases 1 to 30080809)
  AUTHORS   Town,C.D. and Kaul,S.
  TITLE     Direct Submission
  JOURNAL   Submitted (10-JAN-2002) The Institute for Genomic Research, 9712
            Medical Center Dr, Rockville, MD 20850, USA, cdtown@tigr.org
...
FEATURES             Location/Qualifiers
     source          1..30080809
                     /organism="Arabidopsis thaliana"
                     /cultivar="Columbia"
                     /db_xref="taxon:3702"
                     /chromosome="1"
     gene            3631..5697
                     /gene="At1g01010"
                     /note="synonym: T25K16.1; similar to NAC domain protein
                     NAM GB: AAD17313 GI:4325282 from (Arabidopsis thaliana);
                     supported by cDNA: gi_16612276_gb_AF439834 1_AF439834"
     mRNA            join(3631..3913,3996..4276,4486..4605,4706..5095,
                     5174..5326,5439..5697)
                     /gene="At1g01010"
                     /transcript_id="NM_099983.1"
                     /db_xref="GI:18378774"
     CDS             join(3760..3913,3996..4276,4486..4605,4706..5095,
                     5174..5326,5439..5630)
                     /gene="At1g01010"
                     /codon_start=1
                     /protein_id="NP_171609.1"
                     /db_xref="GI:15223276"
```

Figure 3: Sample GenBank file. An excerpt from *Arabidopsis thaliana* chromosome 1. Note the header region, containing locus, date, species, authors, etc, and feature section with gene, mRNA, and CDS features.

The BGLab database schema is shown in Figure 2. Each output file of the parsing program, known as Genome Parser, needs to share its format with one of the tables in the database. Genome Parser will output nine files, given a GenBank file: Identifiers, Descriptors, GeneInfo, GeneBlock, up1500, down500, Exons, Introns, and UTRs. A synopsis of the files can be found in Table 2. The full details of these files can be found in part C of Appendix 1.

Filename	**Attributes**
Identifiers	ID | ID_Type | Gene_ID | Model_Number
Descriptors	Gene_ID | Model_Number | Description | Description_Type
GeneInfo	Gene_ID | Chromosome_Number | Strand | number_of_models | Block_Start | Gene_Start | Gene_Stop | Block_Stop
GeneBlock	Gene_ID | sequence
up1500	Gene_ID | Model_Number | up1500_start | up1500_stop
down500	Gene_ID | Model_Number | down500_Start | down500_stop
Exons	Gene_ID | Model_Number | Exon_Number | Exon_Start | Exon_Stop
Introns	Gene_ID | Model_Number | Intron_Number | Intron_Start | Intron_Stop
UTRs	Gene_ID | Model_Number | UTR | UTR_Start | UTR_Stop

Table 2: Genome Parser output files and attributes

As seen in Table 2, Genome Parser explicitly provides the following information for every gene entry in the database: ID, ID type, Gene ID, Model Number, Description, Description type, Chromosome number, strand, number of models, block start/stop, gene start/stop, sequence, up1500 start/stop, down500 start/stop, exon number, exon start/stop, intron number, intron start/stop, UTR, and UTR start/stop. Genome Parser currently resides on Hydra.Dartmouth.edu.

The goal for the program was to be able to take any GenBank formatted file, and produce consistent output for the database. Thus the software had to be designed taking flexibility and robustness into particular consideration, since not all GenBank files are formatted equally. The data of some species is organized much better than others. The *Arabidopsis* genome, for example, is overall very consistent and thorough. That of *Homo sapiens*, however, is not only incomplete, but devoid of the standardized naming conventions found in other organisms. As it stands, Genome Parser has successfully parsed the five chosen organisms, and we are confident that it can parse any other genomes with minimal additional effort.

Genome Parser exists as a collection of Perl scripts, and is built on top of the BioPerl Perl module. BioPerl (www.bioperl.org) provides several lower level parsing routines to facilitate data extraction from files containing genetic sequences, such as GenBank files. Perl was the obvious choice for this task, and its reputation for “practical extraction” remains (not to mention that BioPerl is only available through Perl).

Genome Parser:
Input: I – the input file array containing GenBank files to parse
Returns: 9 files, formatted to the specification in Appendix 1

Genome Parser(I)
- For each file in I
    - L ← divide file into its loci
    - For each locus in L
        - Store information from mRNA feature
        - Store information from CDS feature
        - Store information from gene feature
- Store information from exon feature // not used often
- Calculate (introns, UTRs, up1500, down500, etc) from the stored information
- For each output file
    - Create file with information from above
- Return the 9 output files

Figure 4: Genome Parser pseudocode. A very general overview of how Genome Parser parses GenBank files

Source Code and User's Guide

The source code for Genome Parser can be found on Hydra at </d7/bglab/GenomeParser/>, as well as on my Senior Thesis CD-ROM. A user's guide is also available for the program (included on the CD); it is titled Genome Parser User's Guide.

IV - VxInsight

Overview

One of the applications used to examine datasets is called VxInsight. It is a commercial application that clusters information using the vector space similarity model and a statistical comparison function <www.cs.sandia.gov/projects/VxInsight.html>. We are using VxInsight to visualize datasets from a genomic perspective; that is, to look at patterns formed by genetic sequences. We have used VxInsight to cluster the up1500 regions of genes within a single genome using motif similarity as the objects of comparison. The statistical function for comparison on the objects can be Pearson's R, R^2, or T test[16].

The application takes an input vector table and generates a 3-dimensional graph of the dataset, with nodes representing the individual vectors (genes). Distance between two genes is inversely proportional to their motif similarity, and so the graph produces tightly connected gene clusters loosely connected with each other. See Figure 5 for a picture of the terrain map and graph of the similarity of *Arabidopsis* up1500 regions using 5-mers and 6-mers, and a comparison method of Pearson's R^2.

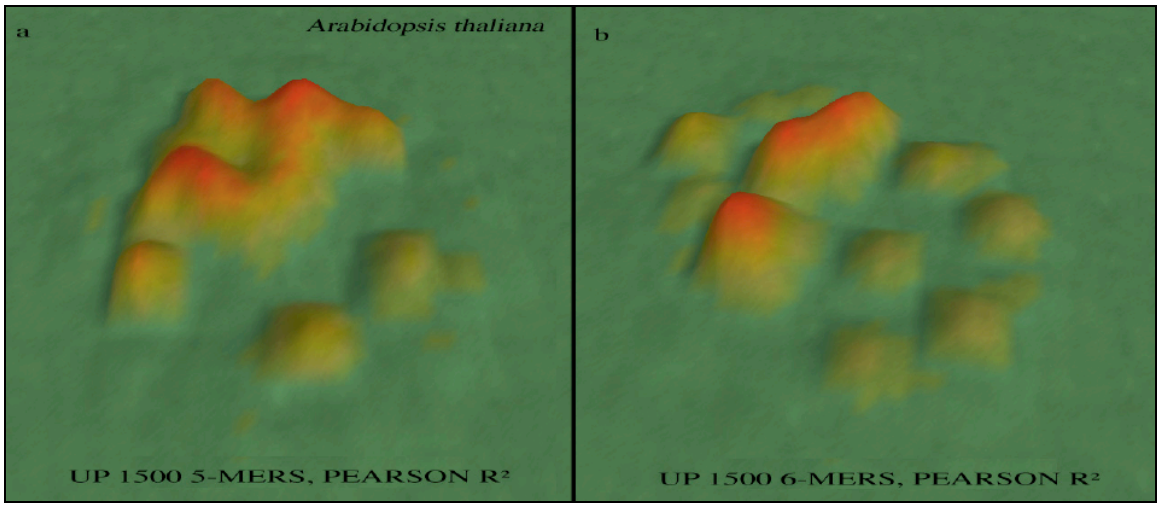

Figure 5: VxInsight results for the up1500 regions of genes from *Arabidopsis*. These results have been generated using vector tables of 5-mers and 6-mers from *Arabidopsis thaliana*, using Pearson's R^2 as the similarity function, with a 0.9 threshold.

Vector Table Creation

To perform this computation, vector tables were created as input. They contain the number of occurrences of an n-mer permutation in a particular gene. The table has rows of gene names, and columns of n-mer permutations. There are currently tables for *Arabidopsis* with n = { 4, 5, 6, 7 }, using Pearson's R and R^2. The tables range in size from 25,498 (the number of genes in the *Ararbidopsis* genome) x 4^4 to 24,498 x 4^7. These tables were generated from the database by Jonathan Carlson, and are the main source of data for the research being done on *Arabidopsis*. (NB the original source of all data is GenBank.) Corresponding tables will serve accordingly for the other four genomes.

Table 3: Sample of the *Arabidopsis* 4-mer vector table

pub locus	AAAA	AAAC	AAAG	AAAT	AACA
At1g01010	40	13	15	27	8
At1g01020	23	13	18	30	10
At1g01030	48	12	17	15	12
At1g01040	36	15	17	20	12
At1g01050	35	12	20	15	15
At1g01060	22	8	6	17	8
At1g01070	45	9	15	25	10
At1g01080	9	9	12	7	7

Similarity Computation and Ordination

With the vector table, VxInsight builds a graph of the relationships between its source data by first measuring the similarity between data items and then generating 2-dimensional coordinates for each gene. It generates similarity values using the vector space model and ordinates using a force directed placement algorithm.

A similarity score (a value between 0.0 and 1.0) is generated for every pair of genes in the vector table. For a table with n genes, there will be roughly $n^2/2$ entries in the similarity table, as the table is symmetric. In this case the motifs of a gene range serve as the unique identifiers for the gene. For example, using the 1,500 bases upstream of a gene, there will be 1,500 motifs. Because we are taking the positions of each motif into account, we do not consider only unique motifs, as is commonly done with other vector space similarity engines, such as Internet search engines. Thus identical motifs that appear more than once in a gene are weighted equally. To determine the similarity score between two genes, you treat each gene as an n-dimensional vector, where n = number of motifs (in this case 1,500). Every gene vector created has the same number of dimensions. The similarity score for two vectors is determined using the similarity comparison function (Pearson's R, R^2, or T test). Pearson's statistics work by computing the cosine of the angle of a given vector pair[17]. For our up1500 regions, this process will generate a table with roughly 1.1 million entries.

Once the similarity table is generated, VxInsight runs its ordination algorithm, VxOrd, a force directed placement algorithm. This algorithm is akin to simulated annealing. It creates an edge-weighted graph using genes as nodes, and the similarity values between gene pairs as edges. To determine final positions for each node, a potential energy term is introduced, where energy is minimized using an iterative approach. This term is a combination of attractive and repulsive forces generated by comparing similarity values. Positions are first expanded, then the expansion rate is reduced (as in simulated annealing), and finally detailed positions are determined by slightly shifting energy values to minimize the chance of a position becoming stuck in a local minimum. The full details of this algorithm can be found in various papers by Davidson, Wylie, Boyack, et al.[18][19][20].

Capabilities

The utility of VxInsight can be great. Most notably the application provides a clustering algorithm that can be used to determine the relative similarity of entries in a dataset, a way to visualize the results, an interface to allow manipulation and examination of the results, and a way to connect with a database to link the results back to their source. Visualization as a tool cannot be overstated; it is an immense asset to people, with sight as our most developed sense. People can keenly recognize patterns, and the application of visualization on datasets is very rewarding. To fully set up VxInsight, we needed to connect the application to our database. This required the creation of files describing the tables and fields in which we are interested, as well as how to display this information in VxInsight.

Keep in mind that the question at hand is always, “What makes this set of genes similar and why.” This can be rephrased to read, “What motifs do the up1500 regions of a set of genes have in common.” With an understanding of the capabilities of VxInsight at hand, and these questions in mind, it is possible to discover many commonalities of the promoter regions of genes. There are two ways of interacting with the results: by selecting a subset of genes in the graph and gathering information about them, and by querying a database for genes with certain properties and visualizing those genes in the graph. This is where the BGLab database returns.

VxInsight defines a parameters file that allows a user to specify database tables and fields to connect with the dataset. The VxInsight graphs have been connected with the database so that it is possible to select a subset of nodes in the graph and send the results to a text file that displays the name of the genes selected, as well as all of the connected attribute values returned from the database for that gene. This retrieval method allows for discovering new patterns in the datasets by quickly and easily filtering out unwanted data. For example it would be interesting to examine the set of genes corresponding to one of the red peaks in Figure 5 (see Figure 6). Querying the database for the selected genes would retrieve the chromosome, location, and functional description of each gene.

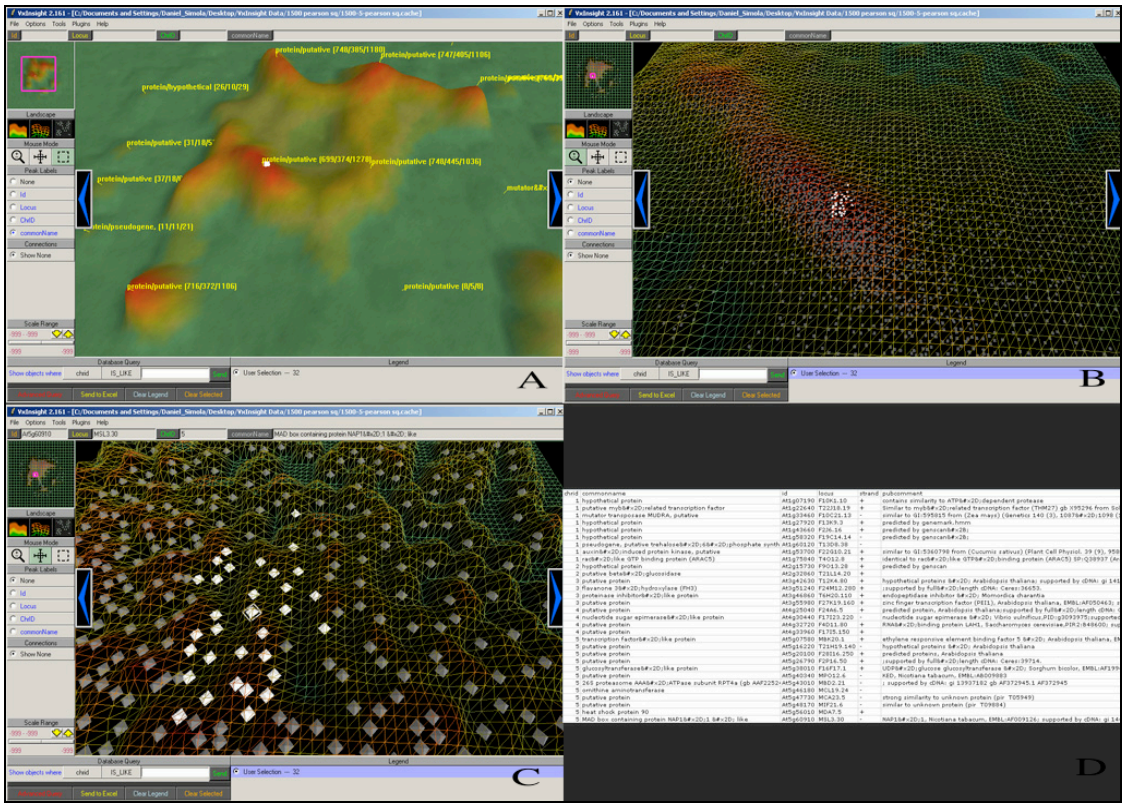

Figure 6: Peak selection from *Arabidopsis* gene cluster. A an example of interactions with VxInsight. (A) Overview of genome using up1,500 regions divided into 5-mers. Same as Figure 5A with the addition of peak descriptions. NB the white selected area near the center. (B) Zoomed view of (A), using wireframe. (C) Additional zooming, with selection of single gene. Notice the descriptions of the gene along top column. (D) Output of the 32 selected genes. Information retrieved from the database.

An alternative approach is to begin with a predefined collection of genes that are related in some way. Using the database interface, you can select genes sharing particular properties, such as location, function, pathway, etc. The first step would be to select these genes in VxInsight. Visualizing their distribution in the graph can provide clues as to the strength of the correlation. Again this application provides a quick way of determining motif correlation using basic pattern recognition. The next step is to determine the specific motifs that correlated the set of genes in the graph. The graph illustrates the existence of correlation, but does not provide the reason. This question inspired the development of another application, called Motif Browser, whose duty is to determine the most representative, or distributed motifs in a set of genes.

V – Motif Browser

Design Goals

Motif Browser is a graphical application, written for Mac OS X in the Objective C programming language. The inner workings of Motif Browser, such as the processes

that actually generate results, are written in Perl. Motif Browser was designed to identify the most representative set of motifs in a set of genes, where representation is defined as appearing either in the most number of genes regardless of frequency in an individual gene, or in the most number of genes with the highest frequency in each gene. We will use the words representative and common interchangeably. It uses the same vector tables as VxInsight, but instead of building a graph from the data, Motif Browser calculates occurrence frequencies. It also connects to the database to determine the positions of each candidate motif for each gene, and provides an output containing the top motifs, the total number of genes in which each motif appears, the frequency of each motif within each gene, and the set of positions of each motif in each gene.

The application aims to illustrate results in four ways: through a standard distribution plot of the location of motifs along the range of a gene (in this case, the up 1500 region), a summary distribution which overlays each individual gene distribution, a histogram which shows the relative frequency of motifs along a gene, and a 'heat map', which uses a color gradient to display the frequency distribution of motifs. Thus Motif Browser provides a parameterized method of explicitly determining the most representative, or common, n-mer DNA sequences in the range of a predefined set of genes, as well as the location of each of these motifs. The motifs returned have the potential to be biologically significant. Motif Browser also answers the question of gene correlation presented in VxInsight.

Determining the Most Common Motifs

The process of determining the most common motifs is rather simple. Let us first call each motif in question a candidate motif. The algorithm begins with a vector table V as in Table 2, the set of genes to consider G (the size of which is g), the number of candidate motifs n the user wants to consider in each gene, and the number of representative motifs m the user wants returned. Let us first clarify the definition of representation with respect to a motif. For some set of genes, a representative motif will appear in the majority of those genes. More specifically a motif is representative if it is in the top m most commonly found motifs for the set of genes in question. Motif Browser allows the user to refine this definition by presenting the ability to look solely at motifs of high frequencies in each gene, or motifs that simply appear in a large number of genes, regardless of their frequency in an individual gene. The motifs returned by Motif Browser are potentially (biologically) significant.

Algorithmically, to find the most representative motifs, you first find the motifs that appear most frequently within each gene in G. From this set you then return the motifs found in the most genes in G. The following describes specifically how the FindCommonMotifs algorithm does this. It first extracts each row from V where a row's gene name matches the name of a gene in G (that is, a gene the user is interested in), and ignores the rest of the table. It sorts each of these rows by motif frequency, from greatest to least. Motif frequency is the value stored in the vector table entry for gene x motif, otherwise known as the column data or attributes of the record. Then for each row it stores the first n motifs and their frequencies, discarding the rest. Now there is a table of the same number of rows, but a smaller number of columns, because it is just considering the top n motifs for each gene. This is called the intermediate table. E.g. if n = 10, m =

10, and g = 15, this table has n*g, or 10 x 15 entries. If the user chooses not to consider the frequency of a motif on a per-gene basis, Motif Browser sets the value n to be 1,500, since this is the maximum number of motifs that could be considered for any up1500 region. An overview of the intermediate table generation can be found in Figure 6.

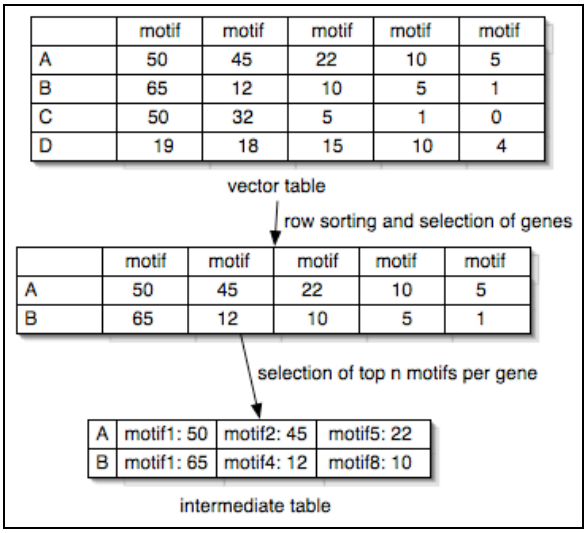

Figure 7: Intermediate table generation

To select the most common motifs from the intermediate table, the algorithm calculates the total number of genes in which each motif appears. It creates a list of every unique motif in the intermediate table, of which there are at most n*g. Then for each motif in the list, it tallies the number of rows (i.e. the number of genes) in which the motif appears in the intermediate table. The number of occurrences of a single motif ranges from 0 to g (a motif has g occurrences when it appears in every gene in the selection. Now there is a list of all unique motifs from the intermediate table, with each motif associated with an integer corresponding to the number of genes in which it appeared. This list is then sorted, and the top m motifs are returned as the most representative of the search. The running time of FindCommonMotifs is $O(V^2)$, determined by the second for loop during the generation of the unique motifs table. Recall V is the vector table. Here it refers to the number of rows in the table. The pseudocode for this algorithm can be found in Figure 8.

This process answers the questions of most frequently occurring motifs across the set of genes being tested and most representative motifs in the set of genes, discounting frequency. Now the question becomes where does each motif lie in each gene of the selection. This necessitates a dataset containing every motif and its location, for every gene in a genome – all information that is stored in the BGLab database. Recall that only the 9-mers are stored in the database. When the database is queried for the positions of a given motif in a given gene, it returns the position of the first base of each matching 9-mer. That is, there is at least one 9-mer matching every other n-mer (where $n \leq 9$). The position of the two matching motifs are the same, since the position recorded for a 9-mer is the position of the first base of the sequence, and only the first n bases of the 9-mer are used in a matching comparison. It is thus simply a matter of querying the database for the positions of each motif in each gene, and the SQL query used is the following:

"Select position from Arabidopsis.up1500motifs m
Join arabidopsis.identifiers i
On m.GeneID = i.GeneID
Where i.ID = <gene> and substr(m.motif, 0, <motif length>) = <motif>
Order by motif",

where Arabidopsis is the current species and up1500motifs and identifiers are the relevant tables. The program now has all of the information it needs to return the results. The results are stored in a file in the format illustrated in Figure 9.

```
FindCommonMotifs:
Inputs:
V - the vector table,
G - the list of genes to consider,
n – the number of top candidate motifs to consider from each gene,
m – the number of candidate motifs to return
Returns: the most common m motifs from the genes in G

FindCommonMotifs(V, G, n, m):
T ← the temp table of genes and motifs
I ← the intermediate table of genes and motifs
U ← the set of unique motifs in I

// first section is the creation of I
• for each row in V
        • if the row's gene is in G
                • add row to T
• for each row in T
        • sort row by motif frequency, from greatest to least
• for each row in T
        • create a new row new_row with top n motifs from row
        • add new_row to I

// second section is the generation of U
• for each row in I
        • add unique motifs to U
• for each motif in U
        • tally the number of rows in I in which the motif appears (total frequency)
• for each motif in U
        • sort motifs by total frequency, from greatest to least
• return the top m motifs from U
```

Figure 8: FindCommonMotifs pseudocode

Visualization

It was not enough simply to generate the counts and positions. The presence of these results begs for new ways to visualize them. Thus Motif Browser offers four graphical representations of the data: a per-gene motif distribution, a summary (all gene) motif distribution, a histogram, and a 'heat map' (Figure 9).

Motif	Total Freq	Gene	Gene Freq	Positions
TTTT	36	At1g49480	55	-675,-1210,-401,-69,-665,-79,-761,-10
TTTT	36	At2g16210	33	-352,-757,-1106,-302,-1112,-435,-138
TTTT	36	At2g24650	19	-193,-561,-264,-41,-1272,-1011,-507,-
TTTT	36	At2g24680	16	-90,-1187,-605,-1167,-1278,-804,-995,
TTTT	36	At2g24690	46	-530,-1215,-554,-54,-564,-494,-1276,-
TTTT	36	At2g24700	60	-1214,-167,-685,-1125,-617,-433,-651,
TTTT	36	At2g35310	31	-1285,-1173,-936,-523,-836,-1370,-10
TTTT	36	At3g06160	16	-335,-1399,-89,-1113,-257,-328,-836,-
TTTT	36	At3g06220	48	-77,-523,-649,-378,-1240,-207,-1200,-
TTTT	36	At3g17010	28	-381,-697,-1445,-258,-156,-507,-1462,
TTTT	36	At3g18960	23	-1329,-231,-544,-326,-724,-563,-797,-
TTTT	36	At3g18990	40	-765,-443,-520,-85,-1247,-1365,-47,-1
TTTT	36	At3g19180	20	-435,-690,-722,-838,-706,-558,-94,-70
TTTT	36	At3g46770	24	-660,-324,-161,-465,-750,-885,-908,-2

Figure 9: Output format for significant motifs algorithm

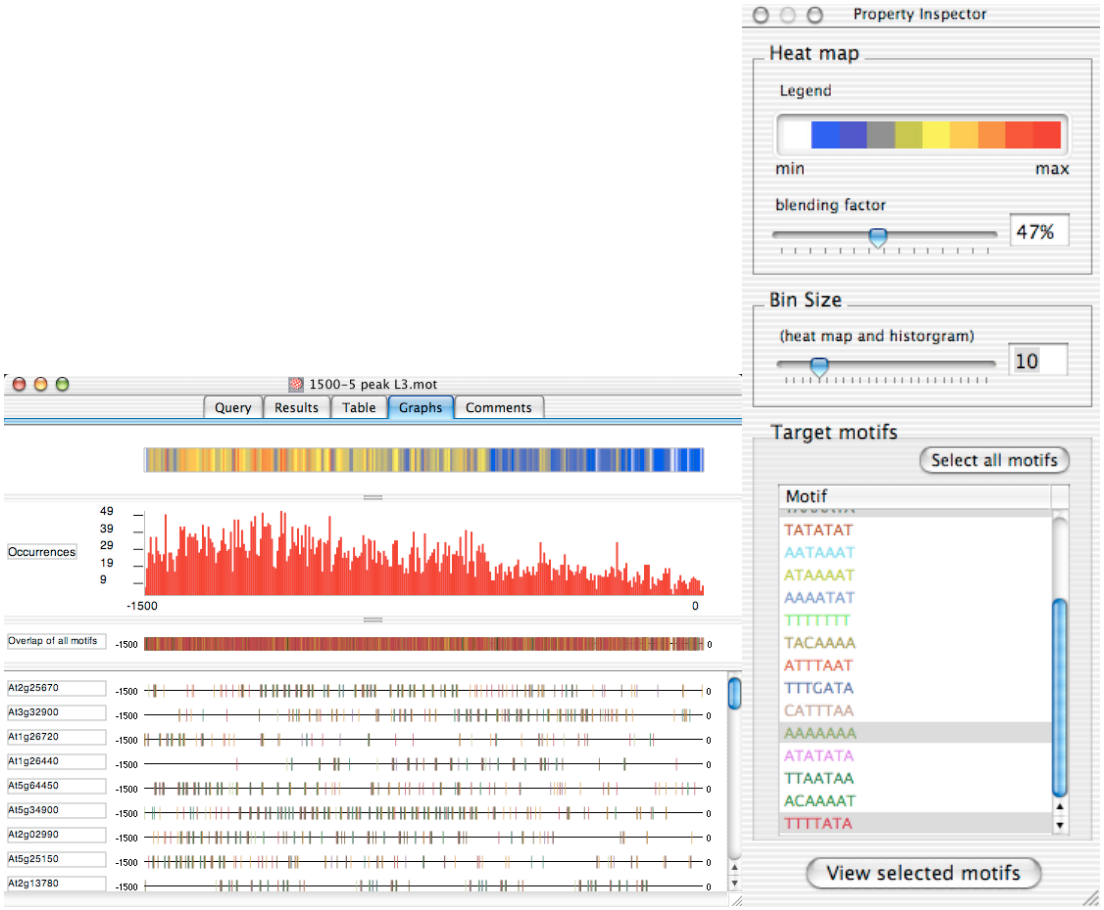

Figure 10: Motif Browser, graphical views and property inspector. Shown on the left are Motif Browser's four views (from top to bottom): heat map, histogram, summary

distribution, per-gene distributions. On the right is the property inspector, showing heat map legend, blending and bin size parameters, and motif selection table.

These graphical representations are provided because they provide something beyond raw data. They allow the viewer to examine and explore the data and extract meaningful patterns visually. These patterns, as in VxInsight, provide the biggest clues for examining motifs. Each representation provides a slightly different way of looking at a set of results, and each view is customizable.

There is a motif selection table in the inspector panel that allows the user to choose a subset of the resulting common motifs to view in the graphics view. This allows the user to narrow focus to examine more selective motif distributions. Every graphical view is updated to reflect only the motifs selected in the inspector.

Per-Gene Distribution

This plot depicts a number line for each gene in the source gene set. The range of each line is the range of the DNA in question: the 1,500 bases upstream of a gene. Thus each range extends from –1,500 to –1. Each line is labeled with a corresponding gene id from the original set. On each line tick marks are drawn corresponding to the position(s) of each motif along each gene. Every gene is always shown on the plot, but only the motifs selected in the inspector panel are plotted. This view allows the user to get a general idea for the layout of motifs along a particular gene, and illustrates the contrast among each gene's distribution.

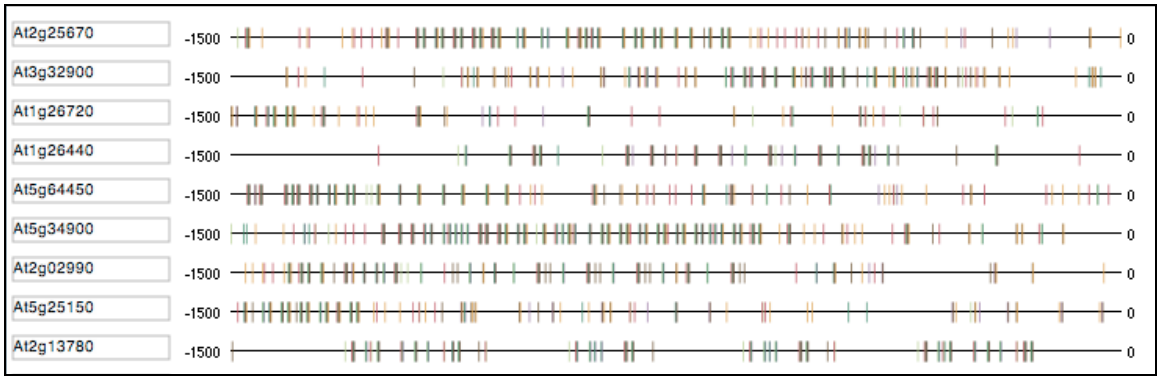

Figure 11: Per-Gene distributions. Each line corresponds to the range of a gene from the source set of genes. Terminal positions are labeled on either side of the line. Tick marks appear on each line corresponding to the positions of each motif in a gene. Each motif is associated with a color for visual distinction.

Summary Distribution

The summary distribution can be described as taking each line in the per-gene distribution and overlapping them. This distribution depicts one line sharing the same range as above (-1,500 to -1). However this one line represents all of the genes in the source set. On the line, tick marks correspond to the positions of each motif across every

gene. The summary line can become very cluttered with motifs, and so it is very useful to use the inspector to view subsets of motifs.

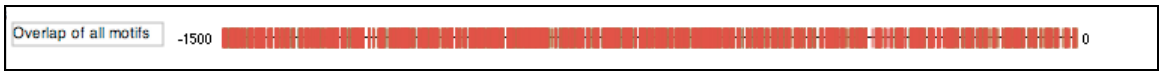

Figure 12: Summary Distribution, showing one line corresponding to the collective range of every gene from the source set of genes. Terminal positions are labeled on either side of the line. Tick marks correspond to the positions of each motif in every gene. Colors correspond to each motif's associated color.

Histogram

The histogram has a domain of bins corresponding to positions along the 1,500 bases of the upstream region, and a range of frequencies corresponding to the number of motifs in a bin. A user can select the bin size from the inspector panel with values from 1 to 50. The bin size corresponds to the number of bases to include in one bin. E.g. if the bin size is 3, then the first bin will include motifs beginning on positions –1,500, -1,499, and –1,498. The height of a bin is the number of motifs found in the bin, and the view is automatically scaled to fit, as the bin size changes.

The histogram offers another way of looking at the motif distributions. Instead of looking at relative motif density as in the summary distribution, the histogram provides the exact number of motifs in a given area along the base range. Changing the bin size is very useful, as the user can look at the overall motif distribution for the results at various levels of resolution.

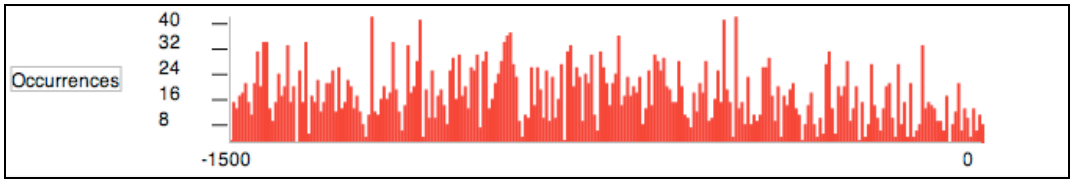

Figure 13: Histogram. The x-axis corresponds to the gene's range (-1,500 to –1), and the y-axis corresponds to the number of motifs in each bin.

Heat Map

The heat map is a map of colors corresponding to frequency ranges from the histogram. For example, white corresponds to 0 motifs in a bin, blue to a low number of motifs, and red to the maximum number of motifs. Hence the name heat map. Think of the heat map as the histogram rotated 90º, so that the user looks down upon the peaks of each bin. The tops of each peak are colored to reflect the relative density of the bin.

This view offers a unique color gradient for each set of results. The user can easily compare the gradients of various sets of results to determine common 'hot spots' of motifs along ranges. The inspector panel presents a blending factor field, which corresponds to the amount that the colors in the heat map 'bleed', producing a blurrier or sharper gradient.

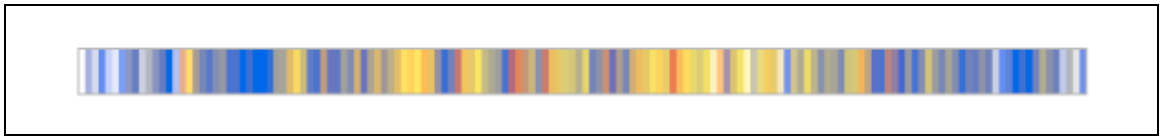

Figure 14: Heat map. Each band corresponds to one bin, and the number of motifs in each bin determines color. A RYB color scheme is used, which goes from white (no data) → blue → yellow → red (max frequency). Bin size is user adjustable.

Choice of Languages

The choice for programming language for Motif Browser was not immediately clear. It could have been written in nearly any language, which in today's field usually means C or Java. The actual FindCommonMotifs algorithm and database querying could have been done in a broader set of programming or scripting languages, or even as a combination of procedural code and SQL queries kept server side. However the choice was made to write the lower level routines in Perl, and build a graphical application on top of them using Cocoa, the Mac OS X development environment.

Again Perl provided the straightforward routines for file handling and parsing. It also provided a relatively painless way of communicating with the BGLab database. The low-level database library code was available through the freetds installation (www.freetds.org), and the Perl CPAN modules, DBI and DBD::Sybase, provided the interface to freetds. DBD::Sybase is a module that operates just above freetds, using the freetds library to handle connections to Sybase databases. DBI, the database independent interface, sits above all of the DBD modules, and provides the programmer with a consistent set of routines to connect to any database. See Figure 15 for an overview of these connections. Instructions for installing this code are available in the Motif Browser installation volume on my Senior Thesis CD-ROM.

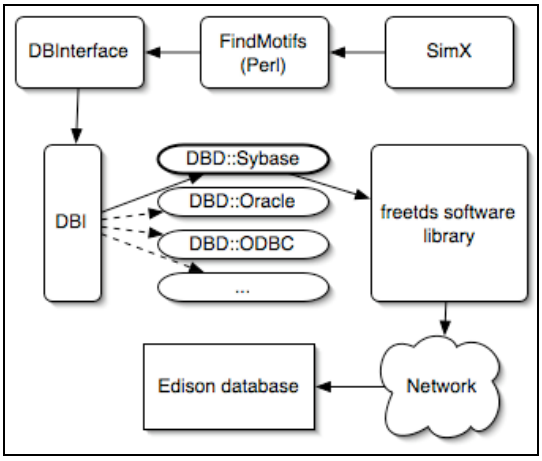

Figure 15: Database connection in Motif Browser

Separating the core functional code from the graphics provides solid scalability. When the lab begins using longer motif sequences, such as the 8 and 9-mers, it will take a long time to calculate result tables. The source vector table for 9-mers for *Arabidopsis* is 25,498 genes x 4^9 motifs. This huge table must be read completely for each computation. The size of the vector tables is also proportional to the number of genes in the selected

genome. Thus there will be (and already is) a need for improved hardware to compensate. In addition to the Edison database server, the BGLab also uses Dartmouth Research Computing's multiprocessor system, Hydra. It is possible to compute all of the results on hydra, and import them into Motif Browser to view graphically. (Results files are small, ranging from a few kilobytes to a couple of megabytes for practical tables.) To import a results file, choose the import item from the File menu of Motif Browser. It will generate a document with the contents of the file, and populate the graphics views.

Also note that the amount of network traffic generated is independent of motif length. At this point the only reason to query the database is to retrieve position information about each motif. This is dependent only on the number of motifs you want returned from the algorithm, and the number of genes to consider. For a set of p genes and m target motifs, there will be m*p queries sent to the database.

The choice to write the application front-end in Objective C using Cocoa was not a difficult choice, as it permits the rapid development of GUIs and functional document-based applications. Apple engineers have shown that allowing a little inefficiency yields a sharp reduction in development time.

Application Design

Motif Browser was designed with the average, non-technical biologist in mind. As a graphically oriented application, it should be easy to use with a clear and simple GUI, but nevertheless powerful in functionality. Graphics can simplify and clarify, but should not be used more than needed. Conversely, odds are that a geneticist will not know how to write SQL queries, so he or she should not be expected to. Thus a delicate balance had to be found, and the goal was to make Motif Browser obvious enough for anyone to use. And, as Buckminster Fuller once said, if the solution isn't beautiful, it isn't right.

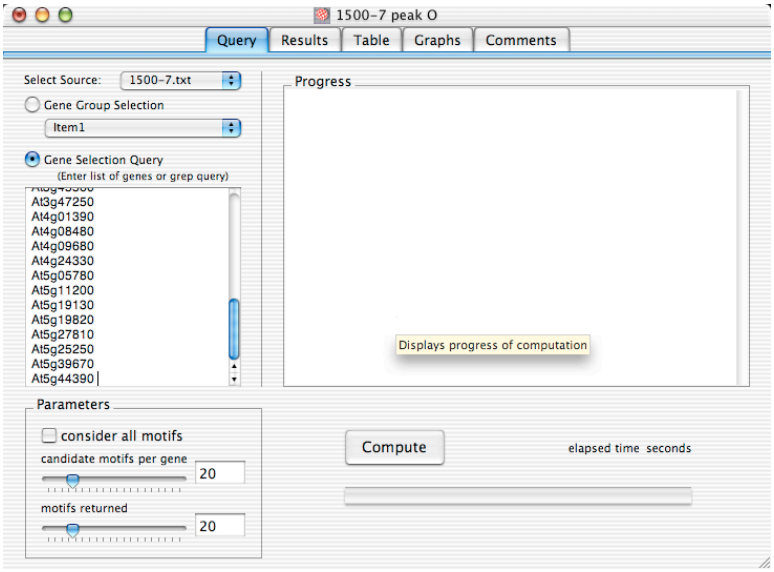

Figure 16: Query view for Motif Browser. On the left hand side, the user is able to select source file, the gene set

(manual or from database), and parameters. The progress window displays the queries being sent to the database.

VI – Results

Using the tools described above, we present preliminary results of interesting motif distributions from the upstream 1,500 promoter regions of selected genes in *Arabidopsis thaliana*. Because the primary focus of this thesis was research-oriented software development, there was little time to actually use the applications to discover and analyze motifs. The results presented are based on 5-mers and 6-mers, and serve as an example of how one would go about searching for sequence elements.

We began in VxInsight, where we clustered the *Arabidopsis* genome using both 5-mer and 6-mer sequences generated from the upstream 1,500 bases of each gene. The correlation function used was Pearson's R^2. VxInsight generated the graph shown in Figure 5. We decided to examine the set of genes from the 5-mer graph comprising the peak shown in Figures 6A-C. We queried the database for the properties of the genes in the selection, and these results are displayed in Appendix 2.

There doesn't seem to be a definitive function or pathway that the 32 genes share, but a few of them are listed as transcription factors. They are also distributed over all five chromosomes. This is an example of the importance of using an updated copy of a genome. More complete annotations could have provided us with more substantial clues about the functions of the genes.

Now that we have a set of genes that appear similar with respect to upstream motif content, it is time to enter the list into Motif Browser to try to identify individual motifs that may be significant. We ran two tests on this set of genes, considering all motifs regardless of their frequency in individual genes. One test used our 5-mer vector table and the other the 6-mer table. Overall results are shown in Figures 17 and 18 below.

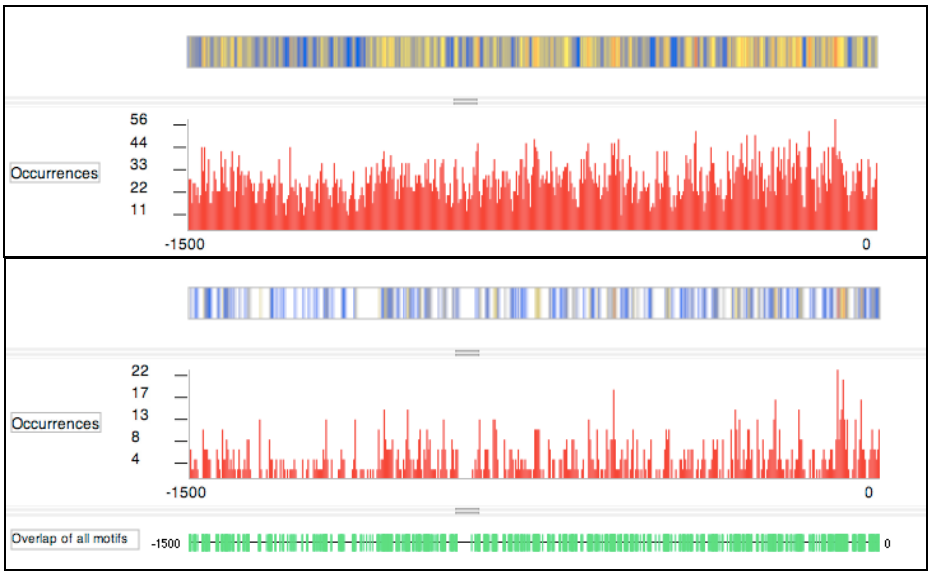

Figure 17: 5-mer distributions using the gene set selected in VxInsight. Figures plotting all motifs returned from Motif

Browser (top). One interesting motif ‘TATAA’, which occurs often in this set of genes. Appears frequently near base -25, suggesting that this motif composes part of the TATA-box of these genes.

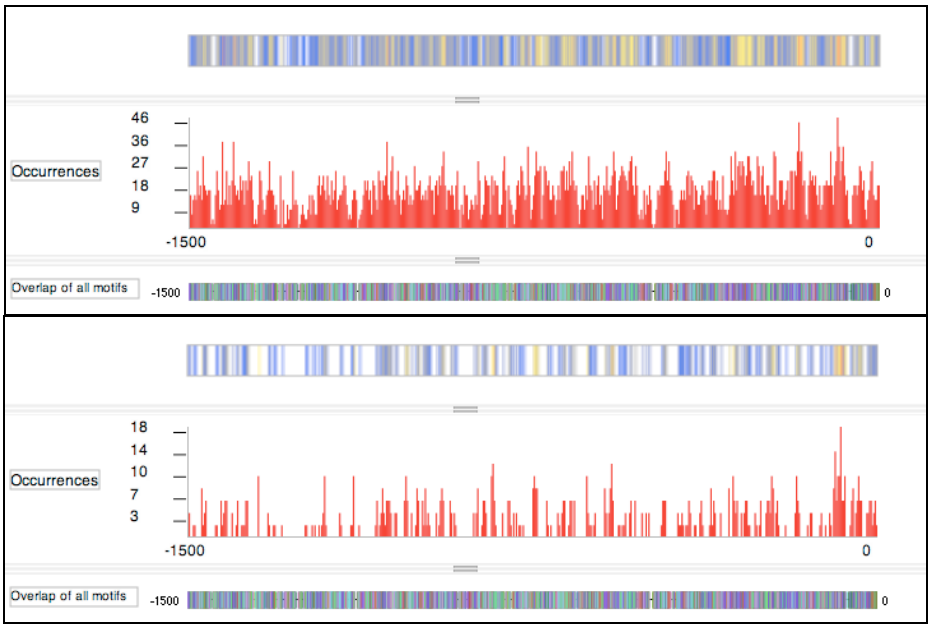

Figure 18: 6-mer distributions using the gene set selected in VxInsight. Overall distribution including all returned motifs (top). Notice the valleys of very few motifs. Distribution of the motif AAAAAA (bottom), which has very selective presence. Appears most often at or near the TATA-box.

Two interesting motifs found in the results were TATAA and AAAAAA, which came from the 5-mer and 6-mer computations, respectively. They both occur the most number of times of any of the individual motifs returned, and their highest frequencies are found close to the -25 base position in the genes. (Other motifs had much more uniform distributions.) This suggests that the motifs may be part of the TATA-box domain. In general the nucleic acid composition of the motifs returned is AT rich (Figure 19). We are not able to draw specific conclusions from this observation, however.

Motif	# of Genes
AAGAA	32
GAAAA	32
ATTTA	32
AATGT	32
TTTCA	32
TAATA	32
TATAA	32
AAACA	32
AAAAG	32
TAAAA	32
TTTCT	32
GATAA	32
TTAAT	32
TAAAT	32
ATCAA	32
AAGTT	32
AAAAC	32
TCTTT	32
AACAA	32
CTTTT	32
AAATA	32

Motif	# of Genes
AAAAAA	32
TTTTGT	32
AAAAAT	32
CAAAAA	32
ATAAAA	32
ATTTTT	32
AAATAA	32
TAAAAA	32
TTTTTT	32
TTTTCT	32
AAAAAC	32
AAAAGA	31
AAAATT	31
GAAAAA	31
TAATTT	31
TTAAAA	31
TTTTTC	31
TTTTTA	31
AATAAA	31
AGAAAA	31
AATTTT	31

Figure 19: Partial list of motifs returned from 5-mer computation (left) and 6-mer computation (right). Generally the motifs appear to be very AT rich.

VII – Conclusions

Conclusions for Preliminary Results

The preliminary results above show one application of the suite of tools that has been developed for the BGLab. There is also a powerful website front-end for our Genomes database that can be used to look at statistics for given motifs. While no specific conclusions can be drawn from the chosen *Arabidopsis* gene cluster, it did appear that we identified the TATA-box region and motifs TATAA and AAAAAA that contributed, at least slightly, to the domain.

The motifs identified using VxInsight and Motif Browser have the potential to be biologically important. VxInsight provides us with a way of looking at entire genomes and quickly selecting correlated gene sets. These sets are good candidates to examine for identifying conserved motifs. This functionality is provided by Motif Browser, which allows us to identify exact motif sequences and their locations. Of course from such results we are to able to determine whether certain motifs indeed play a functional role in gene regulation, or whether genes corresponding to peaks in VxInsight are truly related in an important way.

We have designed this suite of tools to give biologists the ability to ask a broad set of questions about the relevance of motifs in sets of genes. In this thesis we chose to focus on the regions 1,500 bases upstream of genes, but one can investigate any gene region, as long as there is data for it. We hope that these tools will facilitate the discovery of important regulatory motifs, and provide a substantial head start to biologists doing wet-lab research.

Future Work on the Application Suite

Genome Parser

Genome Parser works well at this point, and has been able to parse the five organisms that will be used in the BGLab in the near future. It is important though, that

this software be maintained to comply with updates and changes in the GenBank file format.

VxInsight

VxInsight has been fully connected with our Arabidopsis database, but it is imperative that it be updated to connect with the newer Genomes database. The current limitation is that it cannot use foreign keys in its database configuration file to access tables in the database. Since the Genomes database is configured to use an integer value as the primary key, VxInsight cannot connect the records in our vector tables with attributes from the database. Possible solutions include waiting for an update to VxInsight or altering the vector tables to be indexed by integer value and gene name.

Motif Browser

Applications are never static pieces of software. There are always bugs, improvements, and features to add. Motif Browser works well, but there are a few things it needs to become more generalized and usable:

1. All aspects of the database connection should be user editable. This includes the host address, port, database name, and table name to search in.
2. User should be able to select the gene range to find motifs. These include up1500, upstream, exons, introns, and down500. Unless the Perl scripts are edited, there is no way to alter the current setting of the up1500 table.
3. User should be able to export snapshots of the graphical views or text files of the tables.
4. Most importantly, the speed of the application is hindered by the network connection. Results could be returned about 80% faster if Motif Browser did not have to query the database as often. Two possible solutions are to store position tables locally on the hard drive, and to create a servelet that runs the Perl backend on the database. Both of these approaches would eliminate network traffic.

Future Research

Once the Genomes database is firmly established with the five genomes in place, new datasets can be created using genes from all of the organisms. One example is to compile lists of homologous genes and use the database and Motif Browser to identify potentially conserved motifs that are common among these genes. Cross genome comparisons will undoubtedly provide scientists with new knowledge concerning the amount of DNA conservation among homologous genes and pathways.

Of course the lab will also be able to do research using the individual genomes as well. There is a vast amount of information contained in these five organisms, and there are myriads of questions about DNA sequences and gene regulation that biologists have yet to ask. With this new suite of tools providing easy access to the genomes of five model organisms, biologists will finally be able to ask these questions.

Appendix 1: GenBank Definitions and Genome Parser Specifications

A. GBK format

Unfortunately, the "Standard" GenBank *.gbk format is hardly standard from genome to genome. It's standard only inasmuch as it allows different genomes to store whatever info they want in a way that is somewhat parse-able. Unfortunately, that means different genomes store different information, or the same information using different labels. This section is my attempt to define the way I see *.gbk files storing information. If the files you have appear to significantly differ from this, let me know, as the formats I'm going to have you parse things into are based off this understanding.

A.1 Structure

A.1.1 Heading: It appears to me that the .gbk "standard" of defining Accession, version, and GI numbers in part of a heading for every submitted sequence considers each chromosome to be a submitted sequence. Therefore, we cannot count on this information being available for every gene.

A.1.2 Features: This will include all the genes, as well as chromosome-wide info (base range, chromosome ID, organism, etc. The major subentries of FEATURE are as follows:

A.1.2.1 Gene: This will include the range of the gene (not clear whether it's translation, transcription, or varies depending on the quality of the experimental data); as with all sequences *compliment*(x..y) means it's on the '-' strand.

A.1.2.2 mRNA: Of the form [compliment(]join([<]x1..y1, x2..y2, … , [>]xn..yn). This is defining all the exons of the gene. < implies the 5' end is incomplete. > implies the 3' end is incomplete. Theoretically, if complete, these exons will include the 5'UTR and 3'UTR

A.1.2.3 CDS: Of the form [compliment(]join(x1..y1, x2..y2, … , xn..yn). This is defining the coding sequence. Therefore, it should be identical to the corresponding mRNA definition, with the exception that the first and last exons will be shorter. If

the mRNA definition includes > and <, the two definitions will likely be the same. The difference between mRNA and CDS should define the UTRs, though I haven't been able to find an actual case where they did.

Note: genbank encourages all genomes to define both CDS and mRNA for each gene, but I don't think there's any guarantee that they do.

A.2 Identifiers

Since all the standard identifiers of GI, Accession, and Version are only defined for sequence headings, and these appear to be used only once per chromosome, we're left with no standard for actual gene IDs. Thus, each organism seems to have come up with its own means of identifying the genes. Here are the ones I've seen in a couple organisms:

A.2.1 /gene="xxx": Real creative one here. Looks related to the version number in some organisms, completely different in others. **This looks like the unique identifier that ties mRNA, CDS, and Gene records together.** As such, you'll probably want to use this to keep track of the records, especially those that have multiple mRNA entries for one gene. More on that later.

A.2.2 /note="xxx": This is a real wild card. In Arabidopsis, this appears to be where they store accession.version, but in one version of worms, it appeared to be where they store function information in the form of whole sentences.

A.2.3 /db_exref="YY:xxx": I'm not sure what this is, other than referring to a specific type of ID in another database. The important thing is YY tells you what type of ID it is. Here's a couple examples I saw:

A.2.3.1 /db_exref="PID:xxx": Apparently a protein ID…

A.2.3.2 /db_exref="GI:xxx": Apparently the GI …

A.2.3.3 /db_xref="WormBase:Y74C9A.3": God only knows. Whatever, the wormbase standard ID is. Presumably accession.version?

A.2.4 /protein_id="xxx.y": If present, this is supposed to be accession.version

A.2.5 /product="xxx": The name of the protein this gene encodes. Unfortunately, for some organisms, this is an apparent identifier, while for others it's a multiple word semi-description. Probably depends on how well the proteins of the particular genome have been characterized.

There are probably many others as well, but you get the idea. The point is, an **identifier is something that gives an apparently unique ID to the gene.**

A.3 Other information

There's several other pieces of information that may be kept. This appears to be either functional information, or source information (ie, whether or not the gene is experimentally proved or not). Here's some I've seen:

A.3.1 /Function="xxx": The only clear-cut one of the bunch. This will be a sentence describing what the protein does.

A.3.2 /Note="xxx": Yup, seen it as an identifier, but sometimes it serves the same function as, well, function. In one of the worm examples, it served as function. Hopefully this is an anomaly to the version of the worm I picked up, as it was different in other aspects to the one Bob had.

A.3.3 /experimental=[not_]experimental: I assume it means what it says.

A.3.4 /codon_start=n: presumably the position in the CDS that translation starts. Either that, or it's just identifying the reading frame, though if CDS is truly the translational unit, I don't know why it wouldn't always be 1. Indeed, I never found an instance where it wasn't.

A.3.5 /product="xxx": Unfortunately, sometimes this is something like "homologue of protein X", so I have a hard time calling it an identifier.

A.3.6 /translation="xxxxxxx": This is just the translation from DNA into amino acids. We will ignore this.

Again, there probably others, but the point is, **other information is everything that is not likely to be unique, and generally contains a highly variable length.** The importance of the latter point comes into play with database efficiency issues.

Finally, notice that Identifiers and Other Info is both specific to a Gene, mRNA, or CDS listing.

B. Our Definitions

Based on the "Standard" gbk file definitions, let's make clear how we're going to define certain items.

B.1 Models

The /gene="" identifier appears to be what links CDS, mRNA, and Gene together. In some instances, there appear to be multiple mRNA's with the same gene name. Therefore, barring any revelations to the contrary, lets **define a gene to be made up of one or more models where each model is an mRNA entry with the same ID as a corresponding Gene entry.** The numbers are arbitrary. We'll give them 1, 2, 3, …

B.2 Transcription Unit (TU)

This is the total range of transcription for this gene. Thus, it extends from the start of the most 5' exon of any of the models to the end of the most 3' exon of any of the models. *Hopefully*, this corresponds to the gene entry described in A.1.2.1, but I can't guarantee that.

B.3 UTRs

B.3.1 5'UTR: The range extending from the TU start to the start of the first exon of a given CDS.
B.3.2 3'UTR: The range extending from the end of the last exon of a given CDS to the TU stop site.

I expect these to be defined for only a few genes.

B.3 Promoter (up1500)

This is defined as **the region extending from 1500 base pairs upstream of the start of the first exon for a given model (not CDS, we want to include UTRs) to 1 base pair upstream of this start site.** When we actually go to look at the 9mers, well will pull an additional 8 base pairs off the end to complete the last 9mers that start in the range [-8..-1], but that doesn't concern us with the initial parsing. Thus, if the first exon starts at 10,000 and is on the (+) strand, we define an up1500 region to be 8,500 – 9,999. Remember, models are defined based on the mRNA entries (if possible).

B.4 Trailer (down500)

This is the opposite of a promoter. That is, it's **the region extending from the last base of the last exon for a given model (not CDS, we want to include UTRs), plus 1, to 500 bases downstream.** That is, if the last exon ends with 10,000 and is on the + strand, we want the region 10,001 – 10,500.

B.5 Exons and Introns

All exons and introns will be based on the mRNA entries, unless those do not exist, in which case they'll all be CDS entries. Now there are two possibilities: there is either only one exon, or there is more than one exon. Thus, you'll see either:
B.5.1 mRNA [compliment(]xx..yy This is very unlikely to occur, but if it does, it's the exon. No introns exist.
B.5.2 mRNA [compliment(]join(xx1..yy1, xx2..yy2, …) Most common. Each xx..yy pair is an exon, with xx and yy being included in the definition. The gaps between exons are introns, and don't include xx and yy. Thus, [xx1..yy1] is an exon, and (yy1..xx2) is an intron. There must therefore be one less intron than exon for each model.

B.6 Strand Identification

A sequence is assumed to be on the + strand unless it is enclosed by *complement()*, in which case it's on the – strand. Note that if it is on the – strand, you must take the *reverse compliment* of the sequence, where A→T, C→G, G→C, and T→A, and the sequence is in reverse. Also, all arithmetic operators are reversed. Thus, if you see the sequence *compliment(10000..5000),* the promoter region is 11,500 – 9,992.

B.7 Our unique ID

Thus, we finally come to the ID we'll be giving things. As is apparent by now, an ID is more model-specific than it is gene specific, since things like protein_ID, function, exons/intron definitions, UTRs, promoters, etc may vary by model, and are in fact stored in the *.gbk file under the various models. Therefore, the most intuitive way to define our ID is to be of the form GENE_ID.MODEL_NO; unfortunately, decimals make it difficult to deal with in the database, especially since many (most) "model-specific" information is really going to be redundant between models and the easiest way to get rid of duplicate info is using the UNIQUE command, which won't work well with this decimal format. Therefore, we are going to essentially divide the ID into two fields: GENE_ID | Model_Number. Therefore, **you must create a unique integer ID for each gene encountered, and use this whenever you refer to any aspect of any model of the gene. In addition, you must number the models and include this number in all model-specific files.** These numbers are essentially arbitrary; however, if possible, **it would be great if the ID corresponded to the gene's relative position on the chromosome**. That is, gene i's furthest upstream base will be upstream of gene (i+1)'s furthest upstream base. The genes should already be in this order in the *.gbk file. If not, don't worry about it too much. Also, GENE_ID doesn't take into account chromosome number, so **be sure each GENE's ID is unique for all genes in all chromosomes.**

C. File Formats

Based off this understanding of how the information is currently stored, here's how we'll do the file formats.

C.1 File Standards

C.1.1 File Names:

Each subentry in section C.2 must be submitted as a separate file named *organism_subEntryName.txt*

C.1.2 Format of File Contents

Each file must have attributes separated by tabs and entries separated by newline characters. Every line (including the last) must end with a newline character. Thus, translate anything here in the form attribute1 | attribute 2 | … | attributeN as being what each line in the file should look like, where '|' is replaced with a tab. **If an attribute is null for an entry, leave it null, but keep the surrounding tabs. Also, quotes are taken literally, so don't add any of your own quotes to any entry.**

C.2 File Contents

Alright, here we get to the substance. These are all the files each organism needs to submit, with the specified contents.

C.2.1 Identifiers

ID | ID_Type | Gene_ID | Model_Number

ID = the value of some Identifier from section A.2 (or something that would fit in with those)

ID_Type = The type associated with that ID (See A.2). May be GI, PID, Protein_ID, Note, or anything else. **No db_xref!! See A.2.3**
Gene_ID = the ID you give this gene
Model_Number = the model number you give this

Notes: Some identifiers are found only in the CDS entry or mRNA entry, which is why they must be associated with both a Gene_ID and a Model_Number. You must therefore match each CDS to its corresponding mRNA and pull out all the IDs listed under both headings. If the specific ID is under the Gene heading, just make an entry for each model. Also, for ambiguous types such as "note=", only include it in ID if it looks like your organism uses Note to store ID. Your discretion, but **if there's spaces, it doesn't belong here**.

C.2.2 Descriptors

Gene_ID | Model_Number | Description | Description_Type

Description = Anything that looks like a description; that is, anything that fits in A.3 **except translation info!**
Description_Type = again, see A.3. Whatever type is associated with the description. May be Function, Note, Experimental, Product, Codon_Start, etc
Gene_ID and Model_Number are the same as always.

Notes: Once again, you'll have to glean this info from gene, mRNA, and CDS headings. This poses a serious problem of how to match the same mRNA model with CDS model, as there's no specific info matching the two together except the specific exon info, which may only differ in one exon (see example in section D) (/gene= might serve as a matching field, but that's not guaranteed). Somehow, you need to figure out a way to do this, because the info we want is in both mRNA and CDS entries. The good news is, alternative splicing looks extremely rare.

C.2.3 GeneInfo

Gene_ID | Chromosome_Number | Strand | number_of_models | Block_Start | Gene_Start | Gene_Stop | Block_Stop

Gene_ID = your ID
Chromosome_Number = which chromosome is the gene on?
Strand = (+ or -) Assumed +, unless sequence includes *compliment()*
Number_of_models = number of unique models
Block_Start = the position of the start of the most 5' up1500 region (see B.3) of all the models.
Gene_Start/Gene_Stop = these correspond to the TU start and stop (See B.2)
Block_Stop = the position of the end of the most 3' down500 region (See B.4) of all the models.

Note: When determining if something is "most 5' " or "most 3' ", be sure to take into account which strand the gene is on. For (-) strand, most 5' is the max position, for (+) strand, most 5' is the min position, etc.

C.1.4 GeneBlock
Gene_ID | sequence

Sequence = The bases extending **from Block_Start to Block_Stop + 8**, as described in C.2.3. We need the plus 8 to catch the last 9mers. Don't forget to do reverse compliment on (-) strand genes!

C.1.5 up1500
Gene_ID | Model_Number | up1500_start | up1500_stop

Up1500_start = position of the start of the up1500 region for this model
Up1500_stop = position of the end of the up1500 region, as described in B.3**.** Thus, **|up1500_stop – up1500_start| = 1499.**
C.1.6 down500
Gene_ID | Model_Number | down500_Start | down500_stop

Down500_start = the start of the down500 region (ie, end of last exon+1).
Down500_stop = end of the down500 region as described in B.4. Again, |down500_stop-down500_start| = 499.
C.1.7 Exons
Gene_ID | Model_Number | Exon_Number | Exon_Start | Exon_Stop

Exon_Number = 1, 2, 3, etc, as determined by this exon's position in this model.
C.1.8 Introns
Gene_ID | Model_Number | Intron_Number | Intron_Start | Intron_Stop

(see exons)
C.1.9 UTRs
Gene_ID | Model_Number | UTR | UTR_Start | UTR_Stop

These are the Untranslated regions described in B.3
UTR = 5 or 3 [single digit], corresponding to which UTR this is.

D. File submissions

D.1 Logging in to Andes

You must log in to andes.cs.dartmouth.edu
As of now, everyone has to use the compbio account to log in:
Usr: compbio
Pswd: Exon6211

We'll make a group soon so we can all log in using our own usernames.

D.2 Directories

Our main directory for this project is **/usr/jaa/jaa/compbio/motifs/**
In that directory, there is a subdirectory for each organism: fly, human, worm, yeast. Each of those directories has a directory called chromFiles. The files you create should be dropped off in these directories. That is,
/usr/jaa/jaa/compbio/motifs/[organism]/chromFiles/

Feel free to do the actual parsing on andes. It's a 16 processor machine with 8GB of RAM. For now, if you develop any parsers or anything, please put them in /usr/jaa/jaa/compbio/ This is our base directory which we are allowed to do with as we please. Feel free to upload the genomes and any code you develop into here.

Appendix 2: VxInsight Result Data from Peak Selection of *Arabidopsis* Genes

Appendix 2: VxInsight Result Data from Peak Selection of *Arabidopsis* Genes

chrid	commonname	id	strand	locus
1	hypothetical protein	At1g07190	+	F10K1.10
1	putative myb-related transcription factor	At1g22640	+	T22J18.19
1	mutator transposase MUDRA, putative	At1g33460	-	F10C21.13
1	hypothetical protein	At1g27920	+	F13K9.3
1	hypothetical protein	At1g43660	+	F2J6.16
1	hypothetical protein	At1g58320	-	F19C14.14
1	pseudogene, putative trehalose-6-phosphate synthase	At1g60120	-	T13D8.38
1	auxin-induced protein kinase, putative	At1g53700	+	F22G10.21
1	rac-like GTP binding protein (ARAC5)	At1g75840	+	T4O12.8
2	hypothetical protein	At2g15730	+	F9O13.28
2	putative beta-glucosidase	At2g32860	+	T21L14.20
3	putative protein	At3g42630	+	T12K4.80
3	flavanone 3-hydroxylase (FH3)	At3g51240	+	F24M12.280
3	proteinase inhibitor-like protein	At3g46860	+	T6H20.110
3	putative protein	At3g55980	+	F27K19.160
4	putative protein	At4g25040	+	F24A6.5
4	nucleotide sugar epimerase-like protein	At4g30440	-	F17I23.220
4	putative protein	At4g32720	+	F4D11.80
4	putative protein	At4g33960	+	F17I5.150
5	transcription factor-like protein	At5g07580	+	MBK20.1
5	putative protein	At5g16220	-	T21H19.140
5	putative protein	At5g20100	+	F28I16.250
5	putative protein	At5g26790	+	F2P16.50
5	glucosyltransferase-like protein	At5g38010	+	F16F17.1
5	putative protein	At5g40340	-	MPO12.6
5	26S proteasome AAA-ATPase subunit RPT4a (gb AAF22524.1)	At5g43010	-	MBD2.21
5	ornithine aminotransferase	At5g46180	-	MCL19.24
5	putative protein	At5g47730	-	MCA23.5
5	putative protein	At5g48170	-	MIF21.6
5	heat shock protein 90	At5g56010	+	MDA7.5
5	MAD box containing protein NAP1-1 - like	At5g60910	-	MSL3.30
5	MADS box transcription factor-like	At5g65070	+	F15O5.3

pubcomment
contains similarity to ATP-dependent protease
Similar to myb-related transcription factor (THM27) gb X95296 from Solanum lycopersicum. ESTs gb T42000, gb T04118, gb AA59
similar to GI:595815 from (Zea mays) (Genetics 140 (3), 1087-1098 (1995))
predicted by genemark.hmm
predicted by genscan+
predicted by genscan+

similar to GI:5360798 from (Cucumis sativus) (Plant Cell Physiol. 39 (9), 958-967 (1998))
identical to rac-like GTP-binding protein (ARAC5) SP:Q38937 (Arabidopsis thaliana (Mouse-ear cress)); supported by
predicted by genscan

hypothetical proteins - Arabidopsis thaliana; supported by cDNA: gi 14190372 gb AF378864.1 AF378864
;supported by full-length cDNA: Ceres:36653.
endopeptidase inhibitor - Momordica charantia
zinc finger transcription factor (PEI1), Arabidopsis thaliana, EMBL:AF050463; supported by cDNA: gi 15810486 gb AY056282.1
predicted protein, Arabidopsis thaliana;supported by full-length cDNA: Ceres:26380.
nucleotide sugar epimerase - Vibrio vulnificus,PID:g3093975;supported by full-length cDNA: Ceres:154741.
RNA-binding protein LAH1, Saccharomyces cerevisiae,PIR2:B48600; supported by cDNA: gi 15810396 gb AY056237.1

ethylene responsive element binding factor 5 - Arabidopsis thaliana, EMBL:AB008107; supported by cDNA: gi 15529283 gb AY052
hypothetical proteins - Arabidopsis thaliana
predicted proteins, Arabidopsis thaliana
;supported by full-length cDNA: Ceres:39714.
UDP-glucose glucosyltransferase - Sorghum bicolor, EMBL:AF199453
KED, Nicotiana tabacum, EMBL:AB009883
; supported by cDNA: gi 13937182 gb AF372945.1 AF372945

strong similarity to unknown protein (pir T05949)
similar to unknown protein (pir T09884)

NAP1-1, Nicotiana tabacum, EMBL:AF009126; supported by cDNA: gi 14423383 gb AF386929.1 AF386929

8042, gb AA394757 and gb AA598046 come from this gene; supported by cDNA: gi 3941409 gb AF062859.1 AF062859

cDNA: gi 1293667 gb U52350.1 ATU52350

266.1

References

[1] Benjamin Lewin, Genes VII. (Oxford, Oxford University Press, 2000), 649.
[2] Lewin, 637.
[3] Lewin, 634.
[4] Lewin, 627.
[5] Lewin, 31.
[6] James D. Watson, The Double Helix. (New York: Simon & Schuster, 1996), xv.
[7] The *Arabidopsis* Genome Initiative. "Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*." http://www.nature.com/cgi-taf/DynaPage.taf?file=/nature/journal/v408/n6814/full/408796a0_fs.html&content_filetype=pdf. Accessed 5/15/03.
[8] William Press, et al., Numerical Recipes in C. (Cambridge, Cambridge University Press, 1992). 623.
[9] Scott F. Gilbert, Developmental Biology. (Massachusetts, Sinauer Associates, Inc., 2000), 229.
[10] Gilbert, 251.
[11] Gilbert 76.
[12] *Arabidopsis* data – The *Arabidopsis* Genome Initiative.
Drosophila data - http://www.ncbi.nlm.nih.gov/mapview/map_search.cgi?taxid=7227.
S. cerevisiae data - http://www.sciencemag.org/cgi/content/full/274/5287/546, Dolinski, K., Balakrishnan, R., Christie, K. R., Costanzo, M. C., Dwight, S. S., Engel, S. R., Fisk, D. G., Hirschman, J. E., Hong, E. L., Issel-Tarver, L., Sethuraman, A., Theesfeld, C. L., Binkley, G., Lane, C., Schroeder, M., Dong, S., Weng, S., Andrada, R., Botstein, D., and Cherry, J. M. "Saccharomyces Genome Database" http://genome-www.stanford.edu/Saccharomyces/, May 15, 2003.
Other data – Lewin, 75.
[13] Andreas Baxevanis and Francis Ouellette, Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins. (New York, John Wiley & Sons, Inc., 2001). 20.
[14] Baxevanis, 24.
[15] Baxevanis, 49.
[16] William Press, 616-636.
[17] Davidson, Wylie, and Boyack, "Cluster Stability and the Use of Noise in Interpretation of Clustering." Sandia National Laboratories, http://www.cs.sandia.gov/pubs/index.htm, 2001.
[18] Davidson, Wylie, and Boyack, 2001.
[19] George Davidson et al., "Knowledge Mining with VxInsight: Discovery through Exploration." Sandia National Laboratories, http://www.cs.sandia.gov/pubs/index.htm, 1998.
[20] Boyack, Wylie, and Davidson, "Domain Visualization using VxInsight for Science and Technology Management." Sandia National Laboratories, http://www.cs.sandia.gov/pubs/index.htm, 2002.